

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

Projekt dyplomowy

*Symulacyjny model adaptacyjnego tempomatu*

*A simulation model of adaptive cruise control system*

Autor: *Jakub Burczyk*

Kierunek studiów: Automatyka i Robotyka

Opiekun pracy: *dr inż., Marek Długosz*

Kraków, 2021

Spis treści

[1 Wstęp @TODO 3](#_Toc89292571)

[1.1 Charakterystyka problemu @TODO 3](#_Toc89292572)

[1.2 Cel oraz założenia pracy @TODO 3](#_Toc89292573)

[1.3 Zawartość pracy @TODO 3](#_Toc89292574)

[2 Historia systemów wspomagania 3](#_Toc89292575)

[3 Pojazdy autonomiczne 3](#_Toc89292576)

[3.1 Poziomy autonomiczności według standardu SAE 3](#_Toc89292577)

[3.2 Liderzy rynku 4](#_Toc89292578)

[4 Środowisko symulacyjne – CARLA 4](#_Toc89292579)

[4.1 Czym jest CARLA 4](#_Toc89292580)

[4.2 Możliwości symulatora 4](#_Toc89292581)

[4.3 Interfejs symulatora 5](#_Toc89292582)

[5 Tworzenie środowisk testowych 6](#_Toc89292583)

[5.1 Standard OpenDRIVE 6](#_Toc89292584)

[5.2 Sposób zapisu pliku OpenDRIVE 6](#_Toc89292585)

[5.3 Oprogramowanie do edycji sieci OpenDRIVE 7](#_Toc89292586)

[5.3.1 MathWorks RoadRunner 7](#_Toc89292587)

[5.3.2 OddLOT 8](#_Toc89292588)

[6 Sensory i czujniki 8](#_Toc89292589)

[6.1 Działanie radaru 8](#_Toc89292590)

[6.2 Symulacyjny model radaru 8](#_Toc89292591)

[6.3 Parametry symulowanego czujnika 8](#_Toc89292592)

[6.4 Filtrowanie obiektów tła 8](#_Toc89292593)

[7 Regulatory 9](#_Toc89292594)

[7.1 Przejście na dziedzinę dyskretną 9](#_Toc89292595)

[7.2 Układ regulacji 9](#_Toc89292596)

[8 Strojenie regulatorów 9](#_Toc89292597)

[8.1 Charakterystyki symulowanego pojazdu 9](#_Toc89292598)

[8.2 Dostrajanie tempomatu 9](#_Toc89292599)

[8.2.1 Część klasyczna 9](#_Toc89292600)

[8.2.2 Część adaptacyjna 9](#_Toc89292601)

[9 Implementacja regulacji w języku Python 9](#_Toc89292602)

[10 Bibliografia 9](#_Toc89292603)

# Wstęp @TODO

## Charakterystyka problemu @TODO

@TODO

## Cel oraz założenia pracy @TODO

@TODO

## Zawartość pracy @TODO

@TODO

# Historia systemów wspomagania

# Pojazdy autonomiczne

## Poziomy autonomiczności według standardu SAE[[1]](#footnote-1)

Według standardu SAE J3016\_202104 możemy rozróżnić 6 poziomów autonomiczności pojazdów. Opisują one wpływ kierowcy i systemu na pojazd, warunki pracy funkcji autonomicznych oraz ich zakresy i limity działania.

Poziom 0 – brak funkcji autonomicznych – na tym poziomie nie uświadczymy żadnych funkcji jazdy autonomicznej, nie oznacza to jednak, że nie są zintegrowane systemy asystujące. Funkcje ograniczają się do ostrzeżeń dla kierowcy w razie wykrycia potencjalnie niebezpiecznych sytuacji oraz wspomagań np. awaryjnego hamowania.

Poziom 1 – podstawowe wsparcie kierowcy – pojazd wyposażony jest w jeden z systemów wsparcia: kierowania lub przyśpieszania i hamowania. Na tym poziomie nie można jeszcze mówić o systemach bezpieczeństwa. Oczekuje się, że kierowca utrzymuje pełną koncentrację w trakcie jazdy. Adaptacyjny tempomat zaliczany jest do tego poziomu autonomiczności.

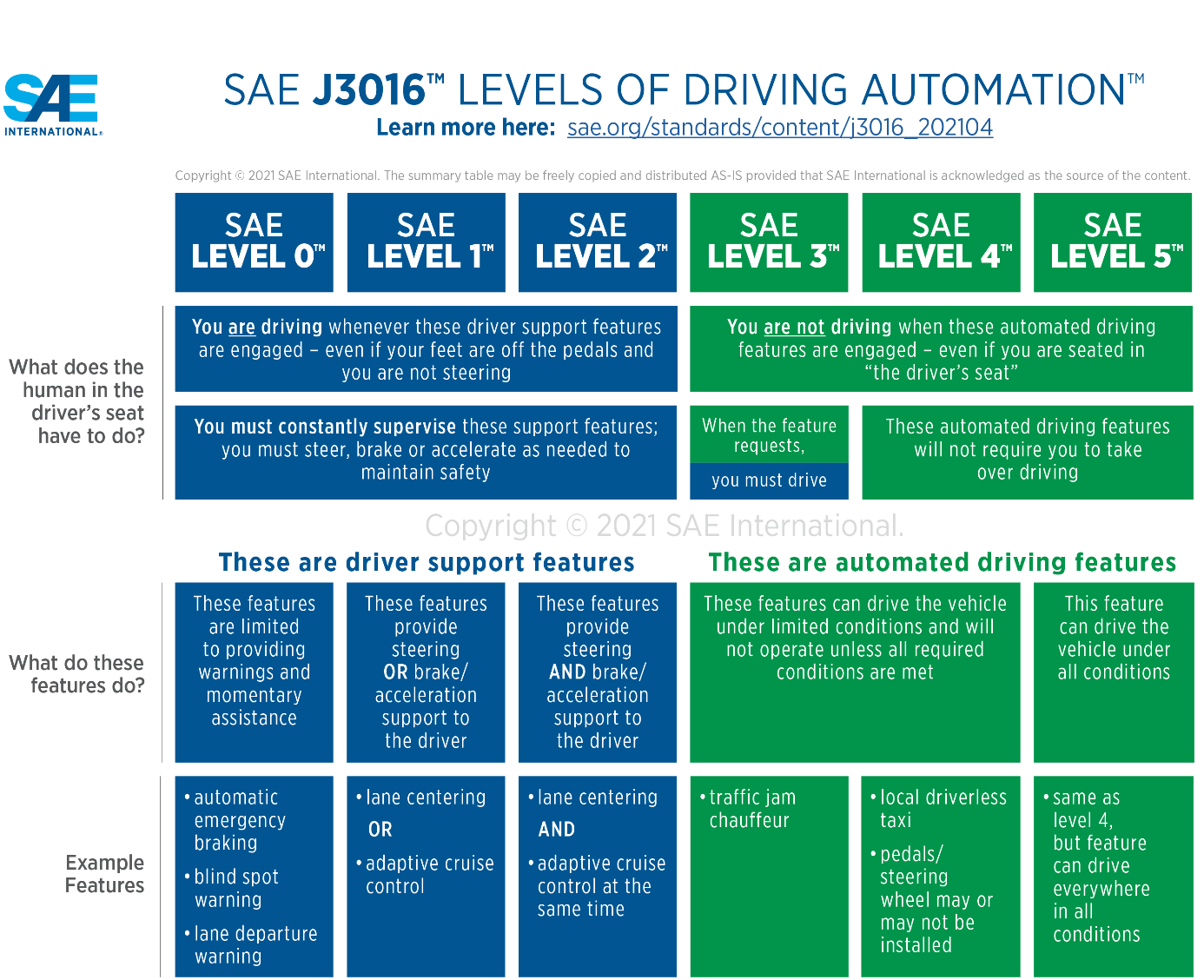
Poziom 2 – częściowa automatyzacja – pojazd wyposażony jest w bardziej zaawansowane systemy wspomagania, które wspierają zarówno sterowanie jak i kontrolę prędkości. Może to być np. połączenie systemu utrzymywania toru i adaptacyjnego tempomatu. Szeroko znany autopilot marki Tesla uznawany jest za system poziomu drugiego. Systemy dalej wymagają ciągłego monitorowania ze strony kierowcy.

Poziom 3 – automatyzacja warunkowa – pomiędzy poziomem 2 a 3 następuje znaczny przeskok możliwości systemu automatyzacji. Pojazd w tej kategorii posiada systemy badania otoczenia i po spełnieniu kryteriów potrafi poruszać się samodzielnie, ale w każdym momencie może zażądać przejęcia kontroli przez kierowcę.

Poziom 4 – wysoka automatyzacja – są to pojazdy o autonomiczności na tyle zaawansowanej, że nie jest konieczne ich przystosowanie do przejęcia kontroli przez człowieka. Są to jednak maszyny o bardziej wyspecjalizowanych zadaniach lub kontrolowanych warunkach pracy jak np. miejskie taksówki.

Poziom 5 – pełna automatyzacja – pojazd jest całkowicie autonomiczny we wszystkich warunkach pracy, a kierowca jest zbędny.

Łatwo można zauważyć wyraźną granicę w tej klasyfikacji. Na pierwszych trzech poziomach kierowca musi zachowywać ostrożność i dalej ma pełnię władzy nad pojazdem Komputer staje się kluczowy dopiero na poziomach od trzeciego wzwyż.

 Rysunek x.x Oficjalny opis poziomów autonomiczności standardu SAE  
źródło: <https://www.sae.org/blog/sae-j3016-update>

## Liderzy rynku

# Środowisko symulacyjne – CARLA

## Czym jest CARLA

Symulator CARLA (Car Learning to Act) jest środowiskiem symulacyjnym zaprojektowanym w celach badawczych i weryfikacyjnych systemów pojazdów autonomicznych. Jedym z celów przyświecających twórcom była całkowita transparentność dla użytkownika i mozliwość rozwoju projektu przez członków społeczności. Dostępny jest on w formie open-source pod licencją MIT[[2]](#footnote-2) zarówno na komputery z systemem Windows jak i Linux. Został zbudowany na bazie silnika Unreal Engine 4, który umożliwia wierną symulację oświetlenia i fizyki obiektów. W celu tworzenia otoczenia wewnątrz symulacji wykorzystywany jest standard OpenDRIVE, za pomocą którego definiowane są sieci dróg oraz ich parametry związane ze sterowaniem ruchem ulicznym.

## Możliwości symulatora

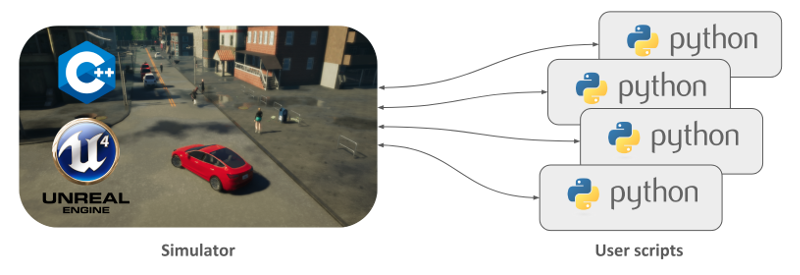
Dzięki zastosowaniu nowoczesnych rozwiązań, wsparciu gigantów technologicznych takich jak Intel, Samsung oraz producentów branży automotive takich jak Mercedes, Toyota czy Valeo symulator zaopatrzono w szereg zaawansowanych funkcjonalności.

Składają się na nie między innymi:

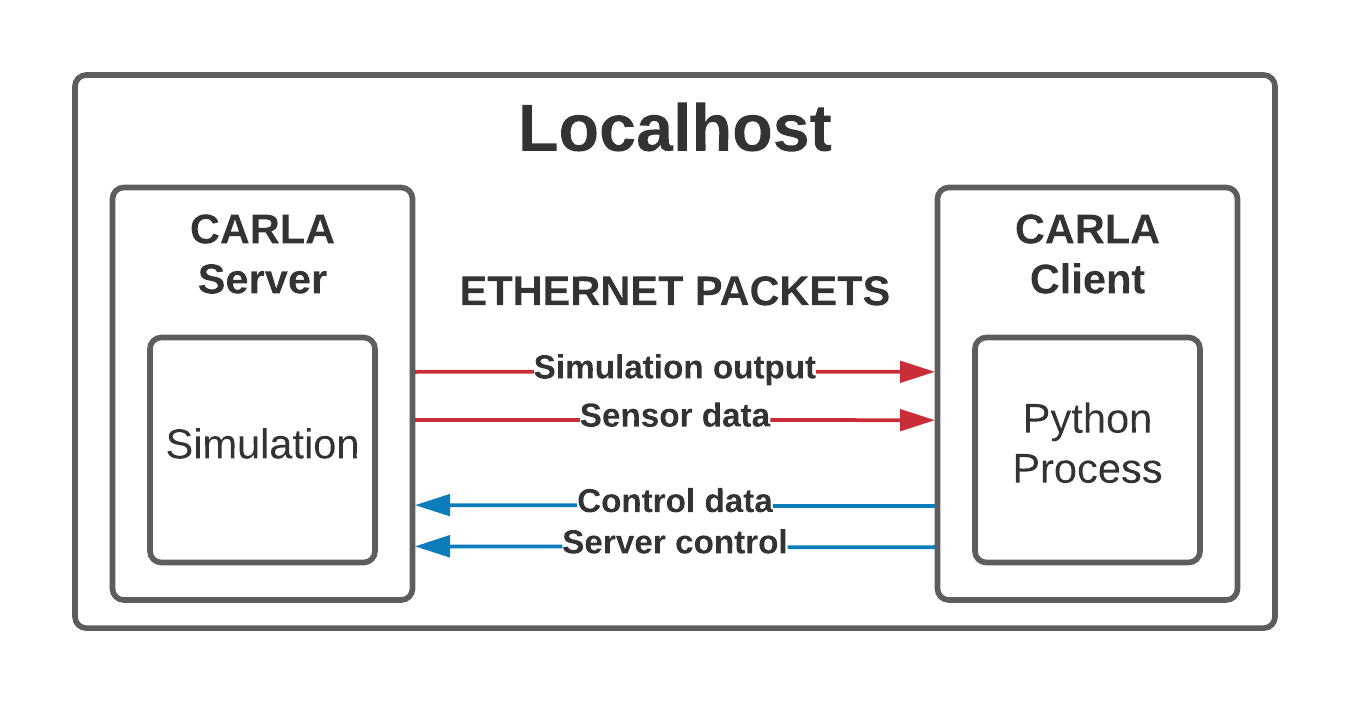
* Symulacja fizyki pojazdów, charakterystyk trakcji, oddziałujących sił i oporów
* Symulacja ruchu ulicznego przestrzegającego przepisów ruchu drogowego, znaków drogowych, sygnalizacji oraz reakcji na innych uczestników
* Symulacja sensorów i pseudosensorów np.:
  + Kamer
    - Głębi
    - RGB
    - Optical Flow – ruch pikseli pomiędzy klatkami
    - Segmentacja semantyczna – podział obrazu na klasy obiektów
  + GNSS – nawigacja satelitarna
  + IMU – jednostka pomiarowa zawierająca akcelerometr, żyroskop i kompas
  + Radar
  + LIDAR
  + Detektory:
    - Detektor kolizji
    - Detektor przekroczenia linii
    - Detektor przeszkód na drodze
* Połączenie z innymi systemami i srodowiskami np. opartymi o platformę ROS[[3]](#footnote-3)
* Tworzenie scenariuszy i środowisk testowych

## Interfejs symulatora

Symulator można podzielić na dwie części, serwerową i poszczególnych klientów. Część serwerowa odpowiedzialna jest między innymi za symulację sensorów, obliczenia fizyki obiektów, aktualizację świata i aktorów[[4]](#footnote-4). Klienci natomiast, za pośrednictwem API[[5]](#footnote-5) w języku Python, mogą komunikować się z serwerem poprzez wysyłanie komend i meta-komend zawierających informacje np. o sterowaniu lub zmianie parametrów symulacji. Obie części symulatora mogą działać równolegle na jednej maszynie, do której odwołujemy się poprzez hosta lokalnego, bądź poprzez sieć komputerową. Umożliwia to uruchomienie serwera na jednostce o dużej mocy obliczeniowej, która udostępnia klientom wysokiej jakości symulację, odciążając tym samym zasoby klientów, które mogą być przeznaczone na algorytmy sterowania.



Rysunek x.x Ogólny schemat architektury symulatora  
źródło: <https://carla.readthedocs.io/en/0.9.12/start_introduction/>



Rysunek x.x Schemat komunikacji w obrębie maszyny lokalnej

Należy zauważyć, że struktura ta może być rozszerzona o dowolną ilość klientów, z czego żaden nie jest limitowany co do ilości i rodzaju symulowanych obiektów.

# Tworzenie środowisk testowych

Jednym z założeń projektowych było stworzenie kilku środowisk, po których mógłby poruszać się symulowany pojazd wyposażony w moduł tempomatu. Choć dostarczone razem z symulatorem mapy są bardzo szczegółowe, a więc dobrze odwzorowujące faktyczne warunki na drogach, to ich poziom złożoności utrudnia kontrolowane testy minimalizujące ilość zmiennych, a dodatkowo cechują się znacząco mniejszą wydajnością. W związku z tym projektowane środowiska będą możliwie minimalistyczne. W tym celu symulator CARLA zostanie uruchomiony w trybie odczytu map OpenDRIVE[[6]](#footnote-6). Po włączeniu symulatora uruchamiany jest odpowiedni skrypt wczytujący plik danych mapy, a następnie automatycznie generowana jest jego geometria w przestrzeni trójwymiarowej.

## Standard OpenDRIVE

Symulator CARLA domyślnie korzysta z otoczeń (map) zbudowanych z obiektów 3D. Pod tą warstwą wizualno–fizyczną znajdują się informacje,które przekazywane są do symulowanych pojazdów, zapisane w otwartym standardzie OpenDRIVE. Są to między innymi dane o połączeniach, skrzyżowaniach, ilości i szerokości pasów ruchu, limitach prędkości i geometrii dróg. Podporządkowanie się do tego standardu pozwala na przenoszenie środowisk testowych pomiędzy wieloma symulatorami. Został on zaadoptowany przez liderów branży takich jak BMW, dSPACE czy Vector Informatik.

## Sposób zapisu pliku OpenDRIVE

Dane zapisywane są jako plik tekstowy języka XML o rozszerzeniu .xodr. Każdy węzeł posiada informacje takie jak jego poprzednik, następca, typ obieku, pozycja i wymiary.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Rysunek x.x Przykładowa droga i fragment pliku .xodr  
źródło: <https://www.asam.net/standards/detail/opendrive/>

Ze względu na charakterystykę tego opisu, bardzo szybko przestaje być on czytelny dla człowieka, dlatego do tworzenia sieci używa się wyspecjalizowanego oprogramowania.

## Oprogramowanie do edycji sieci OpenDRIVE

Duże możliwości opisywanego standardu wymusiły powstanie oprogramowania do edycji wizualnej. Powstało wiele, zarówno profesjonalnych, jak i amatorskich rozwiązań. Podstawowo każde z nich oferuje tworzenie i łączenie odcinków dróg oraz ustalanie skrzyżowań. Oprogramowanie profesjonalne zazwyczaj jest trudniejsze w obsłudze, ale udostępnia o wiele więcej funkcjonalności często związanych z konwersją rzeczywistych map i zdjęć satelitarnych, a także dodawania geometrii dekoracyjnej.

### MathWorks RoadRunner

RoadRunner jest prawdopodobnie najlepszym z dostępnych rozwiązań. Posiada bardzo rozbudowany edytor modeli 3D co pozwala na wierną reprezentację całego otoczenia wraz z mapami ukształtowania terenu, a nie jedynie informacji o drogach. Niestety jest jednym z płatnych edytorów, co wyklucza go z większości hobbystycznych zastosowań.

Obraz zawierający tekst, sprzęt elektroniczny

Opis wygenerowany automatycznie

Rysunek x.x Interfejs edytora RoadRunner  
źródło: [https://www.mathworks.com/products/roadrunner.html#road-and-3d](https://www.mathworks.com/products/roadrunner.html%23road-and-3d)

### OddLOT

# Sensory i czujniki

## Działanie radaru

## Symulacyjny model radaru

## Parametry symulowanego czujnika

## Filtrowanie obiektów tła

# Regulatory

## Przejście na dziedzinę dyskretną

## Układ regulacji

# Strojenie regulatorów

## Charakterystyki symulowanego pojazdu

## Dostrajanie tempomatu

### Część klasyczna

### Część adaptacyjna

# Implementacja regulacji w języku Python

# Bibliografia

1. A. Dosovitskiy, G. Ros, F. Codevilla, A. López, V. Koltun – „CARLA: An Open Urban Driving Simulator”; PMLR 78:1-16
2. Dokumentacja symulatora CARLA (w wersji 0.9.12) dostępna pod adresem: <https://carla.readthedocs.io/en/0.9.12/>
3. Dokumentacja standardu OpenDRIVE dostępna pod adresem: <https://www.asam.net/standards/detail/opendrive/>
4. Standard SAE J3016\_202104 „Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles”

1. SAE International – pierwotnie Society of Automotive Engineers – amerykańska organizacja zrzeszająca inżynierów związanych z branżami motoryzacyjnymi i lotniczymi [↑](#footnote-ref-1)
2. Licencja MIT – rodzaj licencji, który pozwala na dowolne modyfikowanie i dystrybucję danego oprogramowania. [↑](#footnote-ref-2)
3. ROS – Robot Operating System – platforma programistyczna zaprojektowana z myślą o tworzeniu oprogramowania robotów, źródło: <https://www.ros.org/> [↑](#footnote-ref-3)
4. Aktor – w rozumieniu symulatora CARLA, jest instancją obiektu symulacji, może być to między innymi pojazd, sensor, obserwator czy też obiekty związane ze sterowaniem ruchem drogowym [↑](#footnote-ref-4)
5. API – Application Programming Interface – narzędzia programistyczne pozwalające na komunikację z danym programem udostępniającym swój interfejs [↑](#footnote-ref-5)
6. W oficjalnej dokumentacji ta funkcjonalność ma miano „OpenDRIVE standalone mode” [↑](#footnote-ref-6)